REMARKS

In the Office Action, the Examiner rejected the claims under 35 USC §102. Claims 1-34, 36-37, and 39-48 remain pending. The claim rejections are fully traversed below.

Reconsideration of the application is respectfully requested based on the following remarks.

REJECTION OF CLAIMS UNDER 35 USC §102

In the Office Action, the Examiner rejected claims 1-34, 36, 37, and 39-48 under 35 USC §102(e) as being anticipated by "Java TV API Specification," Sun Microsystems, Inc., June 1, 1999. This rejection is fully traversed below.

It is important t note that the Java TV API Specification is marked "Sun Proprietary/Confidential." Therefore, although the reference is dated June 1, 1999, this reference was not published or otherwise publicly available. Accordingly, Applicant respectfully submits that this rejection is improper.

Even if this reference were published, it is important to note that the claimed invention does not appear to be disclosed in the cited Specification. Accordingly, Applicant respectfully requests that the rejection of the claims under 35 USC 102 be withdrawn.

In the Office Action, the Examiner rejected claims 1-30, 34, 36-37, 39-41, and 43-48 under 35 USC §102(e) as being anticipated by Judge et al, U.S. Patent No. 6,430,570, ('Judge' hereinafter). This rejection is fully traversed below.

It is important to note that many of the pending claims enable an application to have some input or control over its own execution according to an application lifecycle. For instance, the application itself may communicate with the application manager managing its execution. In

this manner, the application may control (e.g., initiate or prevent) its own state change. Judge fails to disclose or suggest such a system.

Judge discloses a Java Application Manager for an Embedded Device. See Title. The application manager allows remote control of loading, starting, stopping, unloading, and application state querying of applications on an embedded device. See Abstract. In order to provide these capabilities over a network, the embedded device includes a network interface implementing a network protocol which allows clients to send requests to the Application Manager of the embedded device. The Application Program Interface (API) provides remote devices with the ability to specify the desired management of the device. See col. 4, lines 24-37.

In the most recent Office Action, the Examiner asserts that Applicant's arguments amount to a general allegation that the claims define a patentable invention without specifically pointing out how the language of the claims patentably distinguishes them from the references. Applicant respectfully points out that Applicant has made an attempt to paraphrase the claim language in order to better explain the differences between the references and the claimed invention. However, Applicant has attempted to further point out the claim language below, as the Examiner has suggested.

With respect to claims 1, the application itself can communicate a state change request to request a change in its own state. More particularly, claim 1 recites "receiving a state change request from the application, the state change request indicating a request from the application that an application manager initiate a change in state of the application from a first state to a second state" and "initiating the state change of the application in response to the state change request received from the application when the second state is an allowable state according to a specified set of rules, thereby enabling the application to initiate its own state change via the state change request without human intervention."

For instance, the application itself can communicate that it has decided to terminate and requests that the application enter a destroyed state (e.g., from a loaded state, a paused state, or an active state), as recited in claim 2. As another example, the application can communicate that the application has decided to pause its execution and requests that the application enter paused state (e.g., from the active state), as recited in claim 2.

Similarly, claims 3-4 enable an application to request that its own execution be resumed and its state be changed from paused to active. The application manager may then initiate

execution of the application in response to the resume request. In this manner, the application can request that its own state be changed by the application manager.

Judge fails to disclose "receiving a state change request from the application, the state change request indicating a request from the application that an application manager initiate a change in state of the application from a first state to a second state" and "initiating the state change of the application in response to the state change request received from the application when the second state is an allowable state according to a specified set of rules, thereby enabling the application to initiate its own state change via the state change request without human intervention."

In contrast, col. 9, lines 23-30 and col. 4, lines 24-37 of Judge indicate that such requests may be received from a client to change the state of another application (separate from the client). In fact, the Examiner refers to page 12, 4th paragraph, page 13, 3rd – 4th paragraphs, and more specifically to a "startAppl() request." As the Examiner recognizes in the most recent Office Action, "Judge clearly discloses that a user request from client user application such as client user class ApplClientU 516 to client application Applclient 508 (col. 11, lines 9 – 12 and col. 13, lines 7 – 12)." As disclosed in this text and shown in the corresponding figures, the request is received from client 12, not the application 510, where the client 12 is clearly separate from the application 510. In other words, the request is received from a different application than that for which the state change request is directed. Thus, the state change of the application is managed by the application manager, with no input from the application itself. Rather, a user request initiates the state change of the application. In no manner does Judge disclose or suggest the ability of an application to communicate that it has decided to terminate, pause or resume its own execution without human intervention. Accordingly, Applicant respectfully submits that claims 1-4 are patentable over the cited art.

Claim 5, as previously amended, enables the state of an application to be conditionally changed from a first state to a second state (e.g., terminated). In other words, the request is a conditional request that is conditional upon the application's decision to change from the first state to the second state. Specifically, claim 5, as amended, recites in part,

"requesting a first time that the application change its state from a first state to a second state by sending a request to the application, wherein the request is a conditional request that is conditional upon the application's decision to change from the first state to the second state, thereby enabling the application to allow or prevent its own state change from the first state to the second state in response to the conditional request;

determining whether the application has decided to allow its own state change from the first state to the second state in response to the conditional request by ascertaining whether the application has changed its state from the first state to the second state;"

If the application has not changed its state, the application manager may again request that it change its state from the first state to the second state (e.g., when a predetermined condition is satisfied).

In the Examiner's rejection of claim 5, the Examiner cites col. 13, lines 18-35, of Judge, which indicates that the Application Manager locates the correct application object, then calls the stopAppl() method on the Application. The Application object locates the ApplBase instance by the name passed into stopAppl() and calls its terminate() method. In other words, Judge merely discloses passing the name as a parameter. In other words, the stopAppl() method is called to terminate a particular application. However, the parameter does not indicate whether the termination is conditional or unconditional. Stated another way, the Application Manager of Judge does not enable the application to have any control over whether the application terminates in response to the execution of the stopAppl() method.

In the most recent Office Action, the Examiner correctly asserts that in Judge, the application is changing to a different state according to the parameter that is received. This is not the case with the invention of claim 5. Rather, the claimed invention enables the application to control whether it proceeds with the requested state change. This is called a "conditional request." In other words, the state change request is "conditional upon the application's decision to change from the first state to the second state." As such, the step of "determining whether the application has decided to allow its own state change from the first state to the second state in response to the conditional request by ascertaining whether the application has changed its state from the first state to the second state" is performed. Judge fails to disclose or suggest such an invention. Accordingly, Applicant respectfully asserts that claims 5-9 are allowable over the cited art.

Similarly, claim 10, as previously amended, recites "requesting that the application change its state from a first state to a second state by sending a request to the application, wherein the request is a conditional request that is conditional upon the application's decision to change from the first state to the second state, thereby enabling the application to allow or prevent its own state change from the first state to the second state in response to the conditional request," "determining whether the application has decided to allow its own state change from the first state to the second state in response to the conditional request by ascertaining whether

the application has changed its state from the first state to the second state," and "requesting that the application change its state from the first state to a third state when it is determined that the application has not changed its state from the first state to the second state." Thus, a conditional request is sent, as discussed above with respect to claim 5. The "conditional request" requests a state change from a first state to a second. If the application does not allow the requested state change from the first state to the second state, another request is sent requesting that the application change its state from the first state to the third state. Thus, if the application has not changed its state, the application is requested to change its state from the first state to another, third state from that initially requested (a second state). Judge fails to disclose or suggest such a process. In fact, since a state change is always performed as requested in Judge, it is unnecessary to "determine whether the application has decided to allow its own state change," as recited in claim 10. Accordingly, Applicant respectfully submits that claims 10-14 are patentable over the Judge.

Claim 15 recites "requesting a first time that the application change its state from a first state to a second state by sending a request to the application, wherein the request is a conditional request that is conditional upon the application's decision to change from the first state to the second state, thereby enabling the application to allow or prevent its own state change from the first state to the second state in response to the conditional request," "determining whether the application has decided to allow its own state change from the first state to the second state in response to the conditional request by ascertaining whether the application has changed its state from the first state to the second state," and "requesting a second time that the application change its state from the first state to the second state when it is determined that the application has not changed its state from the first state to the second state and a predetermined condition is satisfied."

Thus, claim 15 also recites a "conditional request" in which the application manager requests a state change from a first state to a second state. If the application does not allow the requested state change from the first state to the second state, a request is sent a second time requesting that the application change its state from the first state to the second state. Thus, if the application has not changed its state, the application is requested again to change its state from the first state to the second state. Judge fails to disclose or suggest such a process. In fact, since a state change is always performed as requested in Judge, it is unnecessary to "determine whether the application has decided to allow its own state change," as recited above. In summary, claim 15 enables two subsequent state change requests to be submitted to an

application when the application decides not to allow the first requested state change. Accordingly, Applicant respectfully submits that claim 15 is patentable over the cited reference. Claims 16-20 depend from claim 15, and are therefore allowable for at least the reasons set forth above. However, these claims recite additional features, as well. For instance, claim 19 recites, "wherein it is determined that the application has not changed its state when the application rejects the requested state change." In other words, the application may not change its state as requested by merely rejecting the requested state change. As another example, claim 20 recites, "wherein it is determined that the application has not changed its state when the application is unable to perform the requested state change." Thus, the application may not change its state because it is unable to perform the requested state change. Judge fails to disclose or suggest such a process. Accordingly, Applicant respectfully submits that claims 15-20 are patentable over the cited reference.

Claim 21, as amended, recites, in relevant part, "a mechanism for enabling the application to at least one of initiate and prevent its own state change from a first one of the plurality of states to a second one of the plurality of states without human intervention." Thus, claim 21 recites a rule-based system for managing execution of an application according to an application lifecycle, which includes a mechanism for enabling the application to at least one of initiate and prevent its own state change from a first one of the plurality of states to a second one of the plurality of states. As set forth above, the cited reference fails to disclose or suggest enabling an application to initiate or prevent its own state change. Accordingly, Applicant respectfully submits that claim 21 is patentable over the cited reference.

Claims 22-33 depend from claim 21, and are therefore allowable for at least the reasons set forth above. Moreover, the dependent claims recite additional features. For instance, claim 27 further recites "an application environment object enabling the application to communicate with the application manager, thereby enabling the application to at least one of initiate its own state change, inform the application manager of the application-initiated state change, prevent its own state change, and inform the application manager that it is preventing its own state change that has been requested by the application manager." As another example, claim 29, as amended, recites "an application environment object that enables the application to communicate a state change of the application to one of the plurality of states to the application manager, wherein the state change has been initiated by the application without human intervention." Thus, claim 29 enables an application to communicate its own state change to the application manager, wherein the state change has been initiated by the application. Judge fails to disclose or suggest the claimed invention.

Similarly, claim 30, as amended, further recites "an application environment object that enables the application to request that the application manager change the current state of the application from a paused state to an active state, thereby enabling the application to initiate its own state change from the paused state to the active state without human intervention." Thus, claim 30 enables an application to request its own state change from the paused state to the active state. As set forth above, Judge fails to disclose or suggest a mechanism for enabling an application to communicate with an application manager, either in the form of informing the application manager of its state change or submitting a request for its state change without human intervention.

Claims 31-33 further recite a display context, which is visible in a first state, and invisible in a second state (e.g., when in the active or paused state as recited in claim 32). More particularly, claim 31 recites "a display manager coupled to the application manager and adapted for managing a display context for each of the applications, the display context being in a first state when the display context is visible and being in a second state when the display context is invisible." Claim 32 depends from claim 31, and further recites "wherein the display context is in the first state when the application is in an active state and in the second state when the application is in a paused state." As set forth above, Judge fails to disclose a display context as claimed. In fact, since the user initiates a state change in Judge, the user need not be notified of whether the application is in the active state or paused state via a display context. Thus, Applicant respectfully submits that Judge neither discloses nor suggests the claimed invention. Accordingly, Applicant respectfully submits that claims 21-33 are patentable over Judge.

Claim 34 is directed to a digital television receiver, which loads an application when an application is present in data stream and executes the application according to an application lifecycle. In addition, claim 34, as amended, further recites a "means for enabling the application to prevent a change in state of the application requested by the application manager or to communicate to the application manager a state change of the application from one of a first set of the plurality of states to one of a second set of the plurality of states, wherein the state change of the application is initiated by the application without human intervention." As set forth above, Judge fails to disclose the claimed invention. More specifically, Judge fails to disclose or suggest an application initiating its own state change without human intervention. Accordingly, Applicant respectfully submits that claim 34 is patentable over the cited art.

Claims 36, 37 and 39 depend from claim 34. More particularly, claim 36, as amended, recites "wherein the second set of the plurality of states includes a paused state indicating that the

application has been paused and a destroyed state indicating that the application has been terminated, thereby enabling the application to pause or terminate its own execution without human intervention." Judge fails to disclose or suggest the ability of an application to pause or terminate its own execution without human intervention. Claim 37, as amended, recites "means for enabling the application to request that the application manager change the state of the application from one of the first set of the plurality of states to one of the second set of the plurality of states, thereby enabling the application to initiate its own state change without human intervention." Thus, claim 37 recites the ability of the application to initiate a change in its own state by requesting that the application manager change its state to a particular state without human intervention. Judge fails to disclose or suggest the ability of an application to initiate its own state change without human intervention. Similarly, claim 39 further recites, "wherein the first state is an active state and the second state is a paused state." Thus, claim 39 enables the application to initiate a change in its state from the active state to the paused state. Judge fails to disclose enabling an application to initiate a change in its own state from the active state to the paused state. Judge fails to disclose or suggest a mechanism for enabling an application to prevent or initiate its own state change, as claimed. Accordingly, Applicant respectfully submits that claims 34-39 and 43 are patentable over Judge.

Claim 40 depends from claim 34, and is therefore patentable for at least the reasons set forth above. In addition, claim 40 further recites "means for enabling the application to communicate to the application manager that the application cannot change its state as the application manager has requested, thereby enabling the application to prevent its own state change." Judge fails to disclose or suggest the ability of an application to communicate to the application manager that it cannot change its state as the application manager has requested. Claim 41 depends from claim 40, and further recites "means for enabling the application to raise a state change exception indicating that the application cannot change its state as the application manager has requested." Judge fails to disclose enabling the application to raise a state change exception indicating that the application cannot change its state as the application manager requested. Accordingly, Applicant respectfully submits that claims 40-41 are patentable over Judge.

Claim 42 depends from claim 34, and is therefore patentable for at least the reasons set forth above. In addition, claim 42 recites "means for enabling the application to raise a state change exception indicating that the application does not want to change its state as the application manager has requested, thereby enabling the application to prevent its own state

change." Judge fails to disclose or suggest enabling the application to raise a state change exception indicating that the application does not want to change its state as the application manager has requested, thereby enabling the application to prevent its own state change. Applicant respectfully asserts that Judge fails to disclose or suggest a mechanism for enabling an application to communicate with an application manager, as claimed, to enable the application to control its own state changes. Specifically, Applicant was unable to identify a state change exception indicating that the application does not want to change its state as the application manager has requested in col. 7, line 6 – col. 8, line 11, 25-36, as cited by the Examiner. Accordingly, Applicant respectfully submits that claim 42 is patentable over Judge.

Claim 44, as amended, recites, in part, "means for creating a class loader associated with each ~~application~~ one of a plurality of applications such that a plurality of class loaders are generated, each of the plurality of class loaders being ~~class loader is~~ associated with ~~the~~ only one of the plurality of applications ~~application~~, the class loader being adapted for loading one or more classes associated with the application" and "means for employing the class loader to load the classes associated with one of the plurality of applications ~~the application~~."

Claims 45-46 depend from claim 44. More particularly, claim 45 recites unloading classes, which may be performed by de-referencing the class loader as recited in claim 46. Claim 46, as amended, depends from claim 54, and further recites "means for de-referencing the class loader by an application manager."

Judge discloses a Java application manager for an embedded device. See Title. The application manager manages applications, where applications are cached in an application cache. See Abstract. The Application Manager unloads a class when requested using the unloadAppl() method or when the memory management handler selects the class to be unloaded as a result of a low- or no-memory condition. See col. 7, lines 27-36. By caching the class, the Application Manager maintains a reference to the class, thereby forcing the Java Virtual Machine not to garbage collect the class. See col. 7, lines 45-48.

Judge describes the Application Manager 24 as a "non-typical network based ClassLoader." See col. 3, lines 62-65. The Application Manager removes references to the objects that are chosen to be unloaded and causes the unreferenced objects to be removed. See col. 9, lines 41-51. In some JVM 22 implementations, the class's ClassLoader object may also have to be unreferenced for the class to be garbage collected. See col. 9, lines 41-51.

While reference is made in Judge to a ClassLoader object, Judge neither discloses nor suggests constructing (e.g., instantiating) a class loader for an application. Moreover, Judge neither discloses nor suggests constructing a class loader for each application to be executed. In other words, although a single class loader/Application Manager is disclosed, this class loader is not associated with an application to be executed. In fact, Judge implies that the Application Manager operates a single class loader for multiple applications. Thus, the Application Manager would not need to create one or multiple class loaders, as it operates independently to remove references to objects, as described above. In fact, FIG. 2 of Judge illustrates a single class loader rather than a plurality of class loaders. Accordingly, Judge teaches away from an application manager that constructs a class loader for each application to be executed.

Moreover, Judge neither discloses nor suggests de-referencing the class loader (e.g., when execution of the application ends or when execution of the application is terminated), as recited in claim 46. Although Judge discloses dereferencing a ClassLoader object, Judge implies that the Class Loader object is associated with the Application Manager, rather than a single application. In other words, Judge neither discloses nor suggests that the Application Manager de-references a class loader associated with a single application. In fact, Judge implies that the Application Manager/class loader is itself de-referenced. Thus, Judge neither discloses nor suggests an application manager dereferencing a class loader (e.g., when execution of the associated application ends or is terminated). Accordingly, Applicant respectfully submits that Judge fails to disclose an application specific class loader as claimed.
Accordingly, Applicant respectfully submits that claims 44-46 are patentable over the cited reference.

Applicant believes that the independent claims and dependent claims are allowable for the reasons previously set forth. The dependent claims depend from one of the independent claims and are therefore patentable over the cited art for at least the same reasons. However, the dependent claims recite additional limitations that further distinguish them from the cited reference. Hence, it is submitted that the dependent claims are patentable over the cited art. The additional limitations recited in the independent claims or the dependent claims are not further discussed as the above-discussed limitations are clearly sufficient to distinguish the claimed invention from the cited art. Thus, it is respectfully requested that the Examiner withdraw the rejection of the claims under 35 USC §102.

It appears from the Office Action that the Examiner intended to reject claims 31-33 and 42 under 35 USC §103 as being unpatentable over Judge et al, U.S. Patent No. 6,430,570, ('Judge' hereinafter), although this rejection was not explicitly set forth. This rejection is fully traversed below.

The Examiner admits that Judge does not disclose the display context being in a first state when the display context is visible and being in a second state when the display context is invisible. Applicant was unable to find a teaching or suggestion of this implementation in the cited reference.

Claims 31-33 are dependent upon claim 21, and therefore are patentable for at least the reasons set forth above. Claims 31-33 further recite a display context, which is visible in a first state, and invisible in a second state (e.g., when in the active or paused state as recited in claim 32). More particularly, claim 31 recites "a display manager coupled to the application manager and adapted for managing a display context for each of the applications, the display context being in a first state when the display context is visible and being in a second state when the display context is invisible." Judge fails to disclose or suggest a display context in a first state when the display context is visible and a second state when the display context is invisible. Claim 32 depends from claim 31, and further recites "wherein the display context is in the first state when the application is in an active state and in the second state when the application is in a paused state." Judge fails to disclose a display context that is visible when the application is in an active state and invisible when the application is in a paused state. Moreover, with respect to claim 33, which depends from claim 31, Judge fails to disclose or suggest an application manager that determines the state of the display context according to one or more rules. Accordingly, Applicant respectfully submits that claims 31-33 are patentable over Judge.

In addition, Applicant respectfully submits that Judge teaches away from claims 31-33. In fact, since the user initiates a state change in Judge, the user need not be notified of whether the application is in the active state or paused state via a display context. Accordingly, Applicant respectfully submits that claims 31-33 are patentable over Judge.

Claim 42 depends from claim 34, and is therefore patentable for at least the reasons set forth above. In addition, claim 42 recites "means for enabling the application to raise a state change exception indicating that the application does not want to change its state as the application manager has requested, thereby enabling the application to prevent its own state change." Thus, the invention of claim 42 enables the application to prevent its own state change

that has been requested by the application manager. Judge fails to disclose or suggest enabling the application to raise a state change exception indicating that the application does not want to change its state as the application manager has requested, thereby enabling the application to prevent its own state change. Applicant respectfully asserts that Judge fails to disclose or suggest a mechanism for enabling an application to communicate with an application manager, as claimed, to enable the application to control its own state changes. Specifically, Applicant was unable to identify a state change exception indicating that the application does not want to change its state as the application manager has requested in col. 7, line 6 – col. 8, line 11, 25-36, as cited by the Examiner. Accordingly, Applicant respectfully submits that claim 42 is patentable over Judge.

If there are any issues remaining which the Examiner believes could be resolved through either a Supplemental Response or an Examiner's Amendment, the Examiner is respectfully requested to contact the undersigned attorney at the telephone number listed below.

Applicants hereby petition for an extension of time which may be required to maintain the pendency of this case, and any required fee for such extension or any further fee required in connection with the filing of this Amendment is to be charged to Deposit Account No. 50-0388 (Order No. SUN1P502).

Respectfully submitted,
BEYER, WEAVER & THOMAS, LLP

Elise R. Heilbrunn
Reg. No. 42,649

BEYER, WEAVER & THOMAS, LLP
P.O. Box 70250
Oakland, CA 94612-0250
(510) 663-1100